ARMY RESEARCH LABORATORY

Reality Check on OpenMP Implementations

by Shirley Moore, Daniel Pressel,
and Juan Carlos Chaves

ARL-TR-2718 April 2002

Army Research Laboratory
Aberdeen Proving Ground, MD 21005-5067

ARL-TR-2718 **April 2002**

Reality Check on OpenMP Implementations

Shirley Moore
University of Tennessee-Knoxville

Daniel Pressel
Computational and Information Sciences Directorate, ARL

Juan Carlos Chaves
HPTi/Major Shared Resource Center, ARL

Abstract

OpenMP is a proposed industry standard Application Programmer Interface (API) that supports shared-memory parallel programming in Fortran and C/C++ on architectures including Unix, Linux, and Windows NT platforms. This report discusses experiences using OpenMP implementations on Shared Resource Center (SRC) platforms. The experiences include running OpenMP benchmarks, as well as using OpenMP with applications. Tools available for debugging and analyzing OpenMP programs are also covered. Most of the results in this report should be considered preliminary and the basis for further investigation.

Acknowledgments

This work was supported by the programming, education, and training (PET) component of the Department of Defense (DOD) High Performance Computing Modernization Program (HPCMP). Additional support was provided by the U.S. Army Research Laboratory-Major Shared Resource Center (ARL-MSRC) and the Common High Performance Computing Software Support Initiative (CHSSI).

This work was made possible through a grant of computing time from the DOD HPCMP and the generous support of several of the Shared Resource Centers, including the ARL-MSRC and the Distributed Center located at the Space and Naval Warfare Systems Center.

INTENTIONALLY LEFT BLANK.

Contents

INTENTIONALLY LEFT BLANK.

List of Figures

INTENTIONALLY LEFT BLANK.

1. Introduction

OpenMP is a proposed industry standard Application Programmer Interface (API) that supports shared-memory parallel programming in Fortran and C/C++ on architectures including Unix, Linux, and Windows NT platforms. Jointly defined by a group of major computer hardware and software vendors who make up the OpenMP Architecture Review Board (ARB), OpenMP is intended to give shared-memory parallel programmers a portable, scalable programming model and simple interface for developing parallel applications for platforms ranging from the desktop to the Supercomputer. (See reference [1] for more information about OpenMP.) OpenMP compilers used here include the following:

- SGI MIPSPro 7.3.1.1 Fortran 77 and Fortran 90 compilers on an SGI Origin 2000 and an SGI Origin 3000 running IRIX 6.5;
- IBM XL 7.1 Fortran 77/90/95 compilers on an IBM Power3 SMP with eight processors per node running AIX 4.3;
- Sun Forte 6 update 1 Fortran 95 on a Sun HPC10000 running Solaris 8; and
- KAI Guide 3.9 Fortran 77 and Fortran 90 compilers on SGI, IBM, and Sun platforms.

2. Benchmark Results

The EPCC OpenMP microbenchmarks are intended to measure the overheads of synchronization and loop scheduling in the OpenMP run-time library [2]. The overhead measurements can be used to compare the efficiency of the run-time libraries of different OpenMP implementations and give guidance on the performance implications of choosing between semantically equivalent directives (e.g., CRITICAL vs. ATOMIC vs. lock routines). Much of these benchmarks address the barrier implementations in OpenMP. However, the overhead itself may not be an indication of how well an individual OpenMP program will perform. An application program will use a whole ensemble of directives, and its performance cannot be predicted on the basis of certain directives alone. However, these benchmarks are meant to give some guidance on choosing directives to the application programmer and give indications to the vendors as to where improvement in their OpenMP implementations may be needed. A detailed explanation of the measurement methodology can be found in reference [2]. A brief explanation is given in this report as follows. The overhead

of a parallel program is defined as Tp-Ts/p, where Tp is the parallel execution time, Ts the serial execution time, and p is the processor count. The overheads of a number of directives are measured in this simple fashion. Overheads are reported in processor clock cycles to allow comparison between different systems.

The loop scheduling benchmark measures overheads for STATIC, DYNAMIC, and GUIDED schedules with different chunk sizes. Results for the Sun E10000, SGI Origin 3000, and IBM Power3 SMP for both vendor and KAI Guide compilers (where possible) are shown in Figures 1 through 5. From these figures, it can be seen that dynamic scheduling is expensive, especially for small chunk sizes. Since the default chunk size is 1 for most OpenMP implementations, users need to be careful to set the chunk size to a larger value when using dynamic scheduling. On the Origin, the overheads of dynamic scheduling are so large as to render it useless, at least with the default setup.

The synchronization benchmark measures synchronization overheads for several barrier types of directives: parallel, for, parallel for, barrier, and single. The overheads of each of the operations are measured for different numbers of threads. Results for the Sun E10000, SGI Origin 3000, and IBM Power3 SMP for both vendor and KAI Guide compilers (where possible) are shown in Figures 6 through 10.

The PBN, or "Programming Baseline for NPB," consists of three sets of source codes based on the NASA Advanced Super (NAS) Computing Division Parallel Benchmark version 2.3. The PBN contains an improved sequential implementation, a sample OpenMP implementation, and a sample HPF implementation. The directives inserted for the OpenMP implementation reflect a programmer's parallelization and data distribution strategy, while the compiler is responsible for implementation and optimization. These benchmarks complement the EPCC benchmarks by providing application-oriented performance measures. Each application in the benchmark has three problem sizes, which are simply called A, B, and C, where C is the largest problem. We looked at the Mflop rate for each problem set as a function of the number of processors. The OpenMP version of the PBN benchmarks has been rewritten by the Real World Computing Program (RWCP)/Omni group in Japan to eliminate some problems. We have not been able to run all size C problems on all of the platforms due to memory limitations and occasional segmentation faults. We will follow up on these problems. Some preliminary results for the Origin 3000 are shown in Figures 11 through 14. These are with STATIC scheduling and the default chunk size. The OpenMP versions of most of the benchmarks appear to scale well for larger problem sizes, although the results shown here are somewhat noisy. We plan to rerun these benchmarks to try to achieve more reliable results and to compare scaling with the (message-passing interface) versions.

3. Lessons Learned

- OpenMP private variables are allocated on a thread's stack. The default stack size may not be large enough for parallel regions with large numbers of private variables or regions that call subroutines with large numbers of local variables which are automatically private. Segmentation faults are a frequent consequence of using a stack size that is too small. Both environment variables and run-time routines may be used to modify the default stack size, although the manner in which this is done is implementation dependent. On the Origin 3000 with the MIPSpro compiler, setting the MP_STACK_OVERFLOW environment variable causes the OpenMP run-time system to automatically detect and report stack overflow errors at run-time. The MP_SLAVE_STACKSIZE variable or the MP_SET_SLAVE_STACKSIZE library routine can be used to request larger stack sizes. Similar facilities are available with some other OpenMP compilers (e.g., KMP_STACKSIZE with Guide).
- Hewlet-Packard currently does not support OpenMP for C. Its support of OpenMP for Fortran is incomplete and less than perfect. In particular, its error messages are extremely poor and generally only state that an internal error has occurred. This behavior insinuates that the compiler is broken when, in fact, it could be a bug in the application code.

Two examples of these problems are the following:

(1) HP Fortran 77 and Fortran 90 support a limited number of continuation lines. Since the limitation applies to compiler directives as well, a problem can arise if there are a large number of private variables. Unfortunately, rather than stating what the problem is, the compiler just gives the internal error message.

(2) The compiler did not seem to work well with code generated by a KAI tool that converted SGI directives to OpenMP. When it was specified that variables should default to SHARED, matters improved. Most of the error messages disappeared, and the job seemed to run correctly.

- On IBM systems, there is a problem that if an OpenMP job tries to use all of the processors, then it is competing with the operating system for the attention of a processor. OpenMP jobs tend to be relatively fine-grained; thus, if the operating system needs 5% of a processor's attention, then the other processors will spend 5% of their time spinning while waiting for the last thread to catch up. Obviously, the problem gets worse as the number of processors in the system increases because the number of processors sitting at a spin lock increases, while the amount of time required by the

operating system can also increase. On the IBM SP, problems are even worse, such as the following:

(1) When performing mixed-mode programming with MPI going between nodes, servicing MPI requests from other processors will also require the attention of a processor, slowing things down even more.

(2) Asynchronous transfer of data between nodes can also put a strain on the memory system, which, in the case of some configurations, is already stretched fairly thin.

- KAI's implementation of OpenMP is based on Pthreads. As such, it should add extra overhead relative to a native implementation. However, our benchmark results so far do not show this to be a problem, and sometimes the KAI compiler outperforms the vendor compiler.

4. Tools for OpenMP

The TotalView debugger from Etnus provides facilities for debugging OpenMP programs as well as for mixed MPI and OpenMP programs [3]. TotalView is available for a large number of platforms and is installed on some Shared Resource Center (SRC) machines. The previous version (4.1) had some problems debugging threaded programs (such as OpenMP) on some platforms (such as SGI), but this problem appears to have been fixed in version 5. TotalView works with both vendor and KAI OpenMP compilers.

The KAI KAPPro toolset includes the Guide compiler, the Assure debugger, and the GuideView performance analysis tool for OpenMP, which are described as follows:

- Guide is a cross-platform implementation of OpenMP for C, C++, and Fortran.
- The Assure component of the KAP/Pro toolset validates the correctness of parallel OpenMP programs and identifies programming errors that occurred when parallelizing a sequential application. The inputs to Assure are an OpenMP parallel program that is assumed to run correctly in sequential mode and a data set for that program. When the Assure-processed program is run, Assure simulates parallel execution and identifies errors where the parallel program is inconsistent with the corresponding sequential program. Assure can display its results using the AssureView graphical user interface or a command-line interface.
- The GuideView component provides an instrumented run-time library that captures timing information for detecting and diagnosing performance

problems in OpenMP parallel programs. The graphical interface provides browsing through performance data to identify parallel regions or loops that require attention.

Performance Application Programming Interface (PAPI) is a specification and reference implementation of a cross-platform library interface to hardware counters [4, 5]. These counters exist as a small set of registers that count "events," which are occurrences of specific signals, and states related to the processor's function. Monitoring these events facilitates correlation between the structure of source/object code and the efficiency of the mapping of that code to the underlying architecture. This correlation has a variety of uses in performance analysis and tuning. PAPI virtualizes the counters on a per-process and per-thread basis and can be used for analysis of threaded programs including OpenMP. PAPI is being installed on some SRC machines.

Vampir is a performance analysis tool for MPI parallel programs developed by Pallas in Germany. Vampir is available on some SRC machines. The next version of VAMPIR will support OpenMP in addition to MPI. Pallas and Intel/KAI are developing a new performance analysis toolset for combined MPI and OpenMP programming which uses PAPI to access the hardware performance counters. PAPI's standard performance metrics, which include metrics for shared memory processors (SMPs), will provide accurate and relevant performance data for the clustered SMP environments targeted by the new tool set.

5. Conclusions and Future Work

OpenMP implementations have matured and will continue to do so. Implementations of OpenMP 2.0 for Fortran will hopefully begin to appear soon. OpenMP is becoming a viable option for scalable parallel programming on shared-memory platforms. We plan to continue our benchmarking work and will investigate possible solutions to performance problems encountered on various platforms.

For example, when using the C$doacross directives on SGI, sometimes the optimal solution is to specify INTERLEAVE, which is equivalent to STATIC scheduling with a CHUNK SIZE of 1. Alternatively, sometimes the optimal solution will be to specify STATIC and let the CHUNK SIZE default. In this case, the default is not 1, rather it is the largest CHUNK SIZE that will result in a uniform distribution of work among the processors (within the limitations of integer division). We plan to investigate use of this optimization with the EPCC scheduling benchmark.

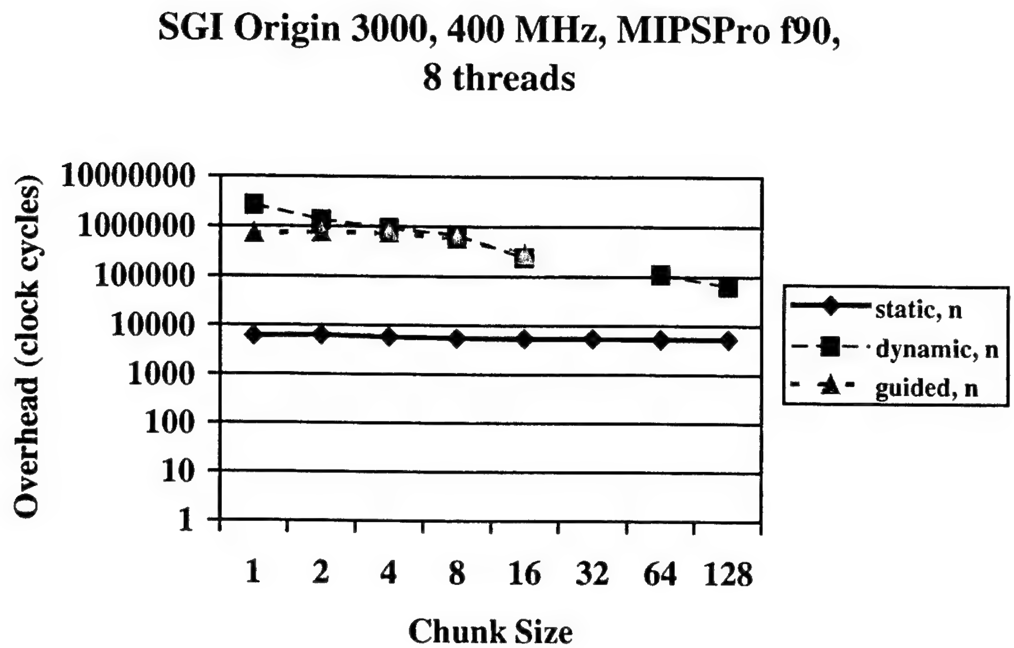

Figure 1. Scheduling overheads on an SGI Origin 3000 with the vendor compiler.

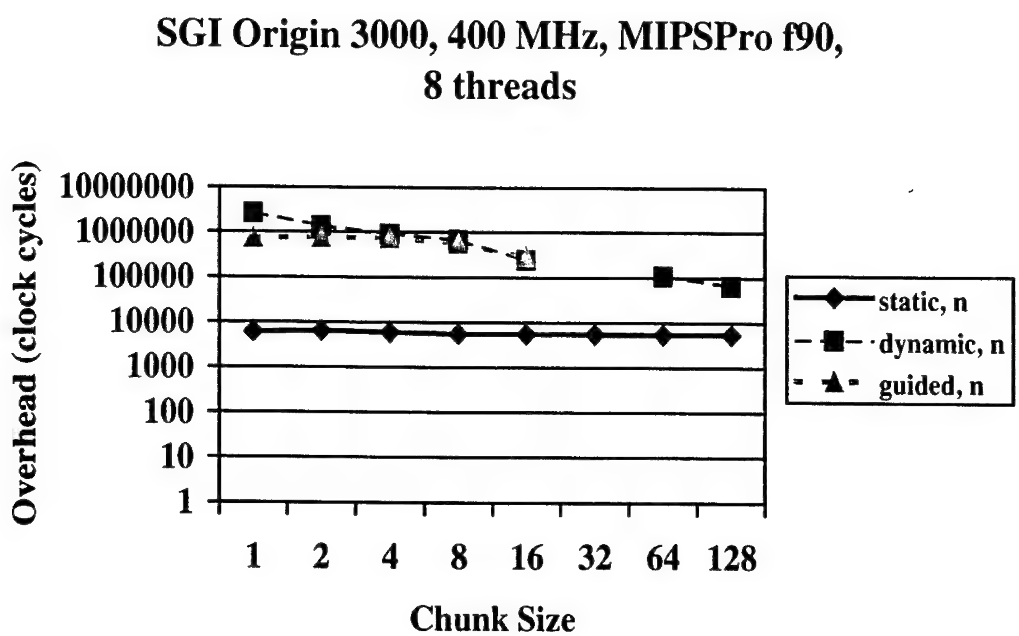

Figure 2. Scheduling overheads on an SGI Origin 3000 with the Guide compiler.

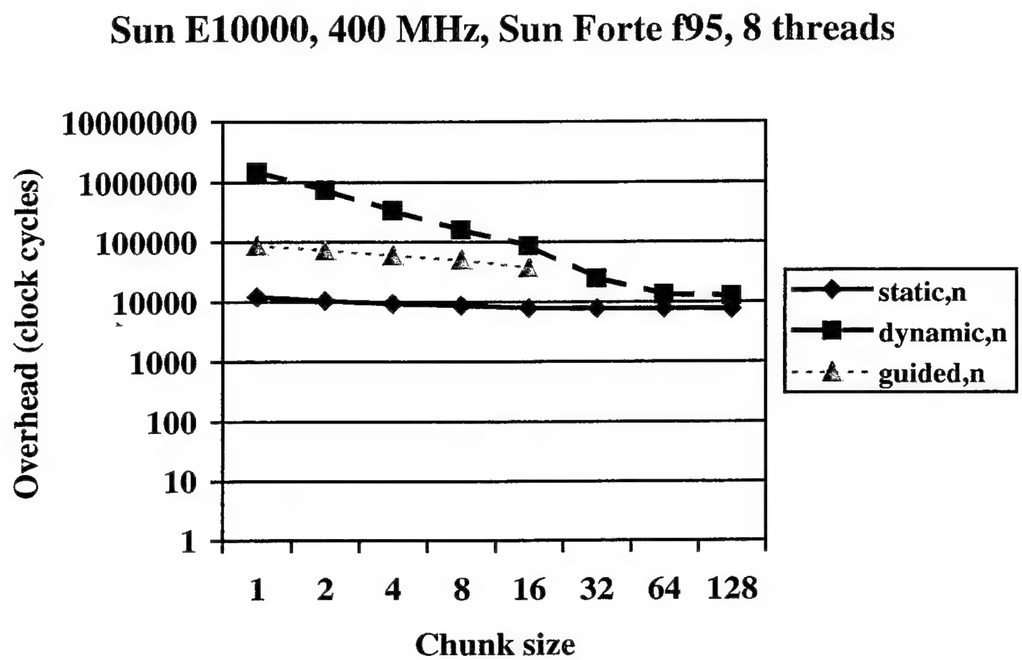

Figure 3. Scheduling overheads on a Sun E10000 with the vendor compiler.

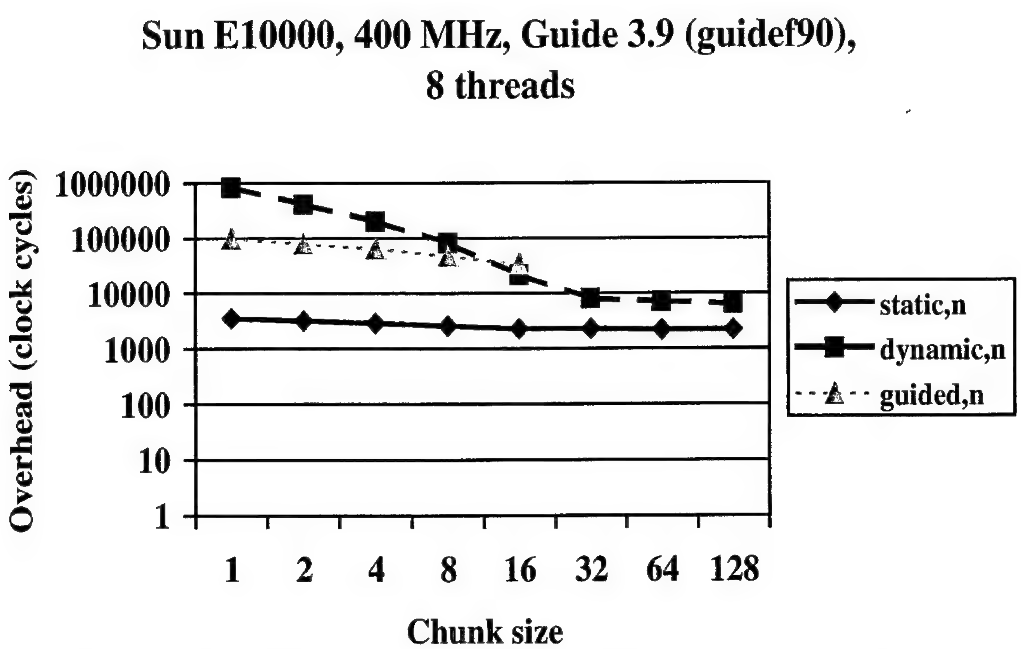

Figure 4. Scheduling overheads on a Sun E10000 with the Guide compiler.

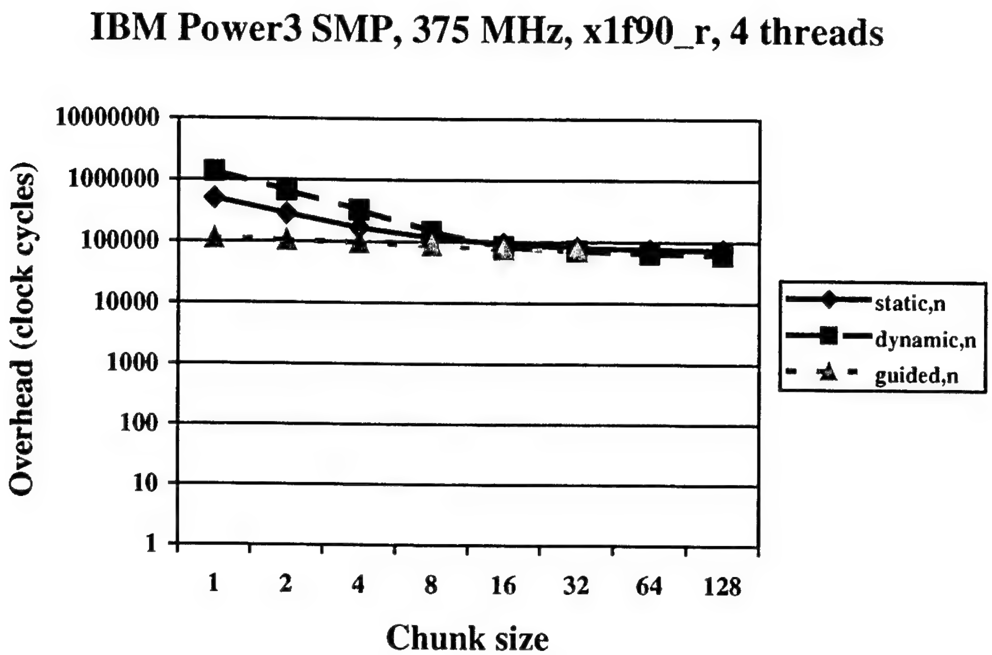

Figure 5. Scheduling overheads on an IBM Power3 SMP with the vendor compiler.

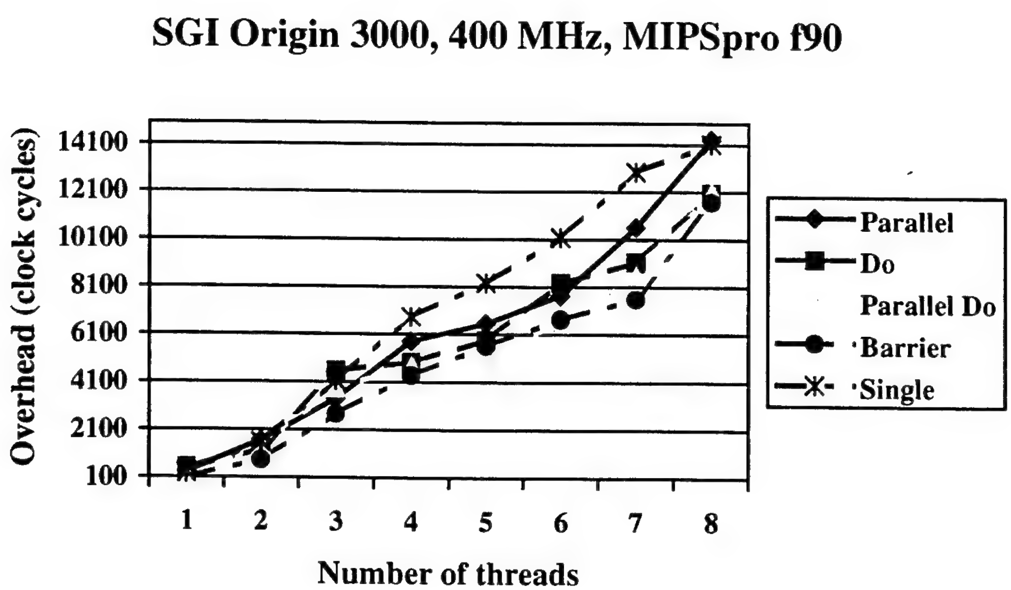

Figure 6. Synchronization overheads on an SGI Origin 3000 with the vendor compiler.

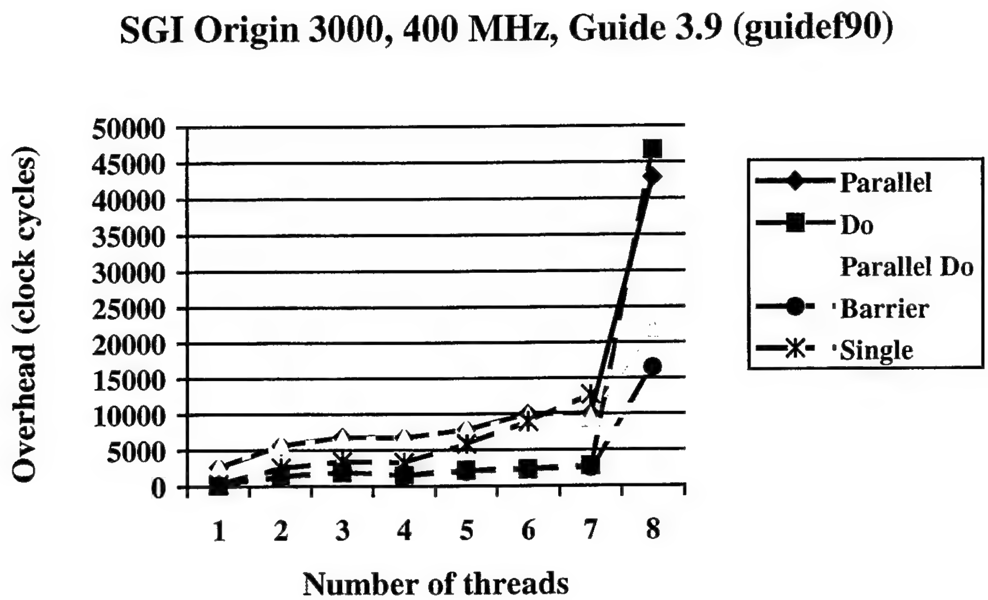

Figure 7. Synchronization overheads on an SGI Origin 3000 with the Guide compiler.

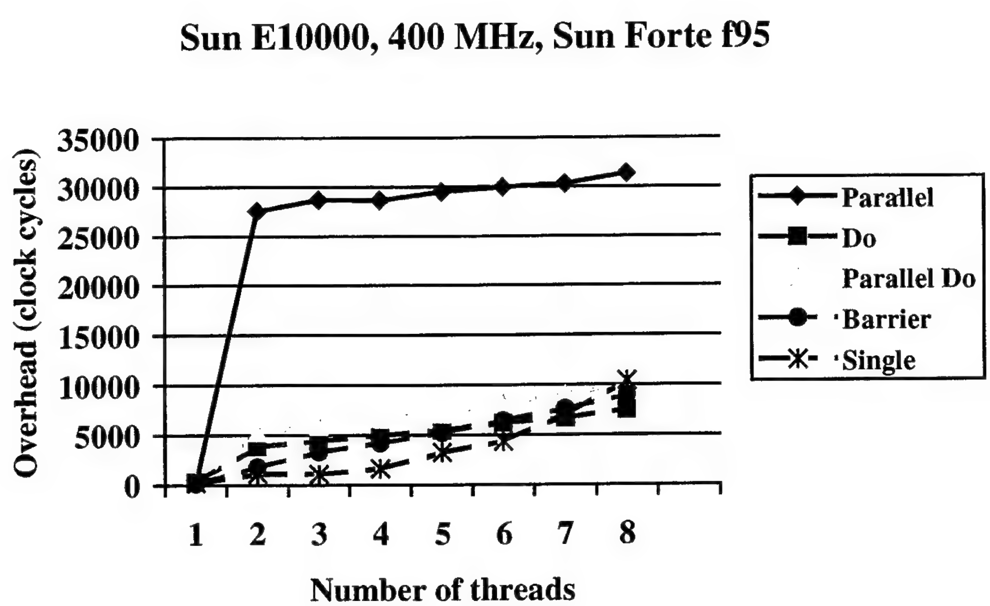

Figure 8. Synchronization overheads on a Sun E10000 with the vendor compiler.

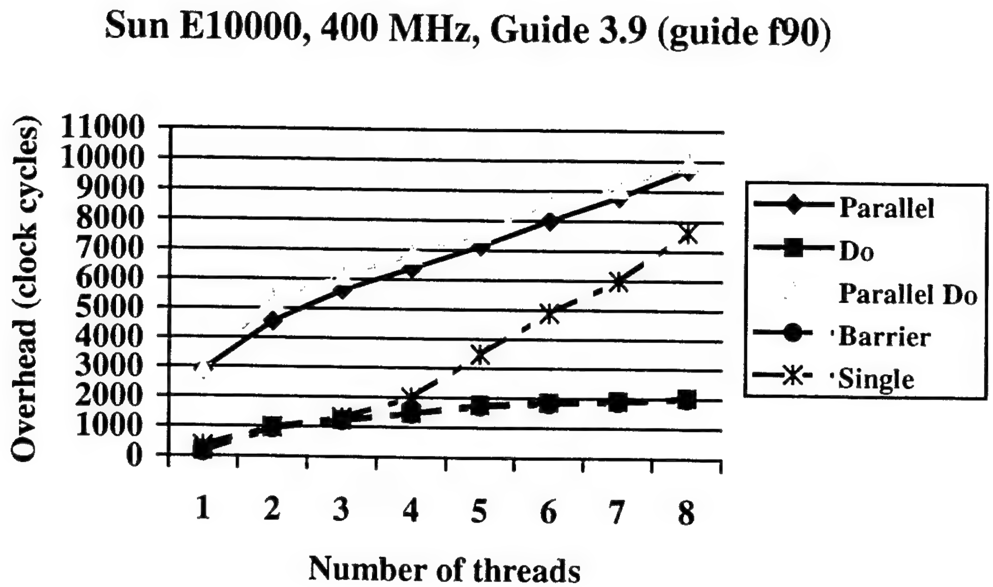

Figure 9. Synchronization overheads on a Sun E10000 with the Guide compiler.

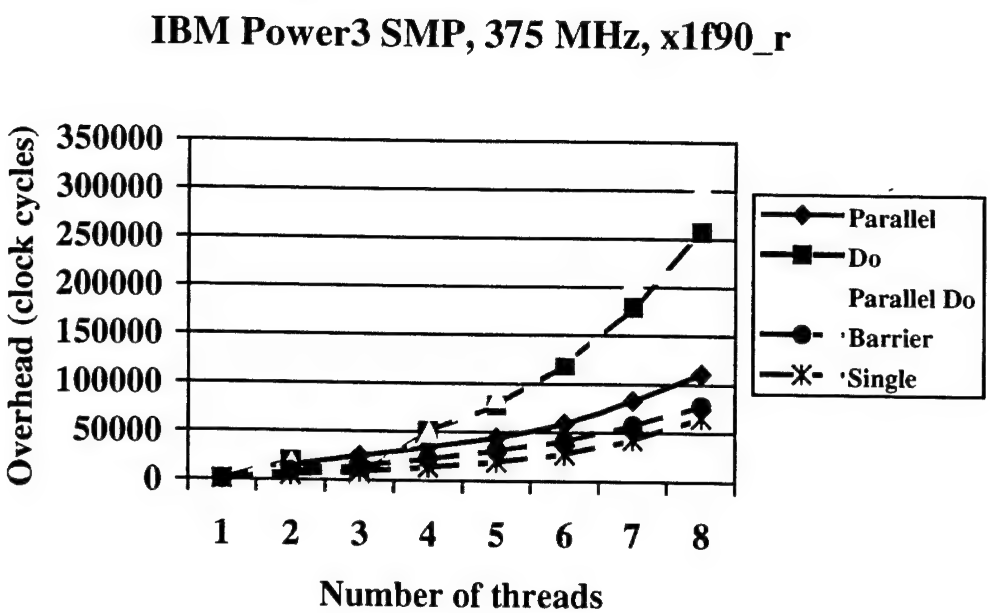

Figure 10. Synchronization overheads on an IBM Power3 SMP with the vendor compiler.

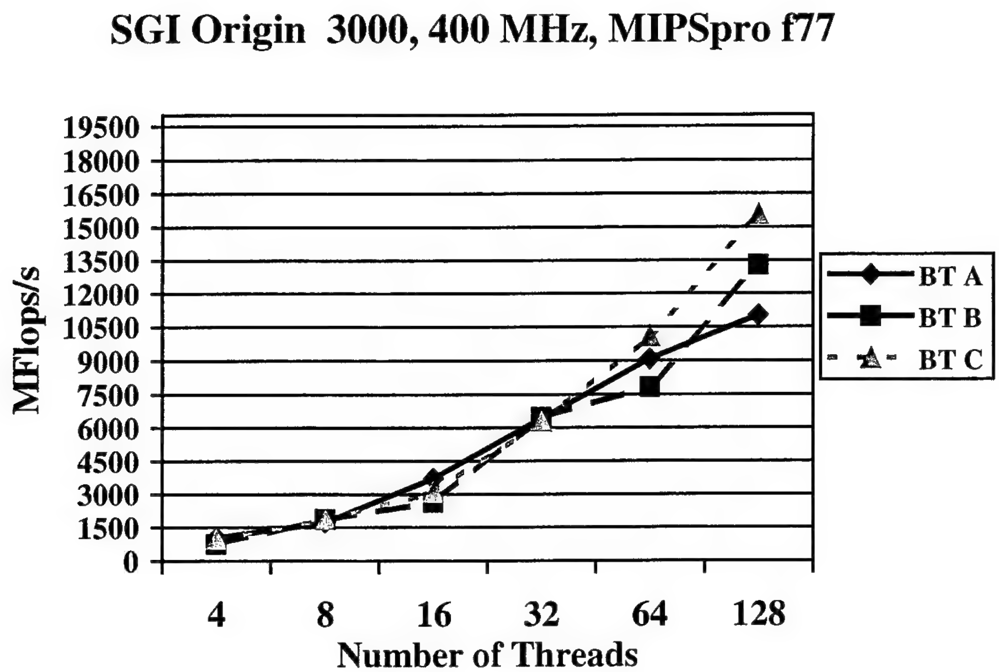

Figure 11. PBN BT benchmark on an SGI Origin 3000 with the vendor compiler.

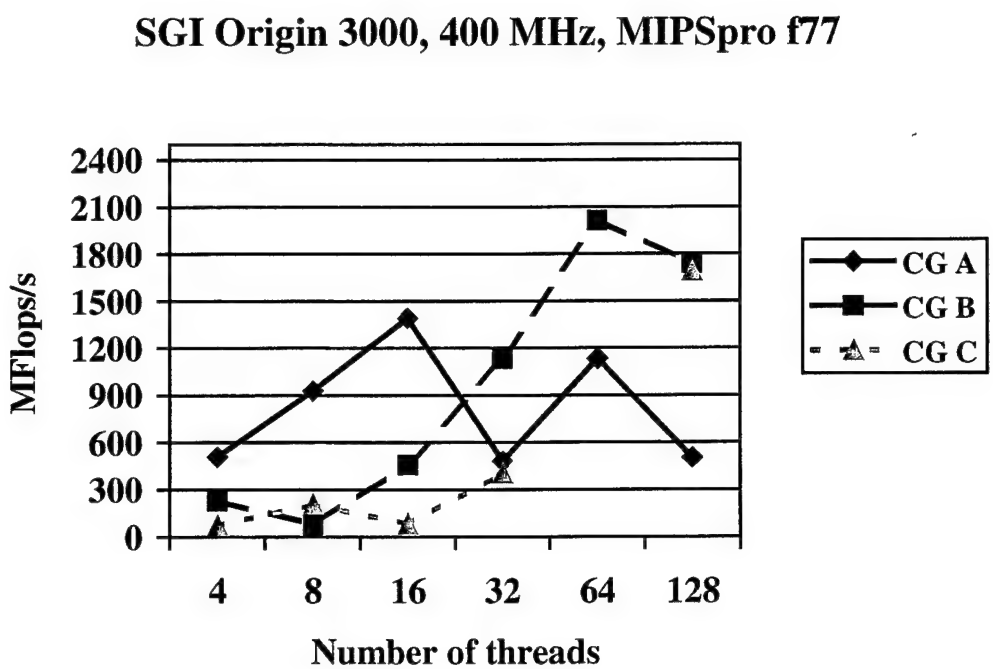

Figure 12. PBN CG benchmark on an SGI Origin 3000 with the vendor compiler.

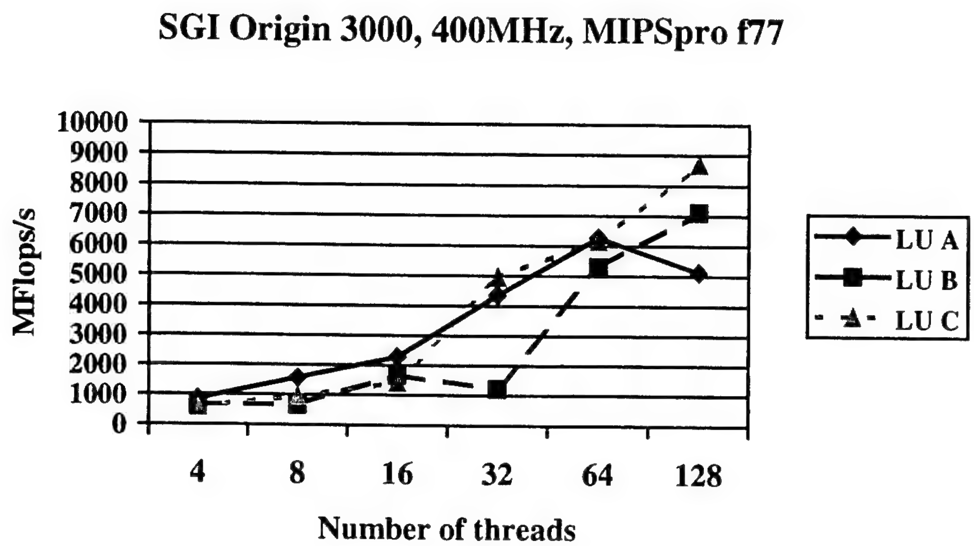

Figure 13. PBN LU benchmark on an SGI Origin 3000 with the vendor compiler.

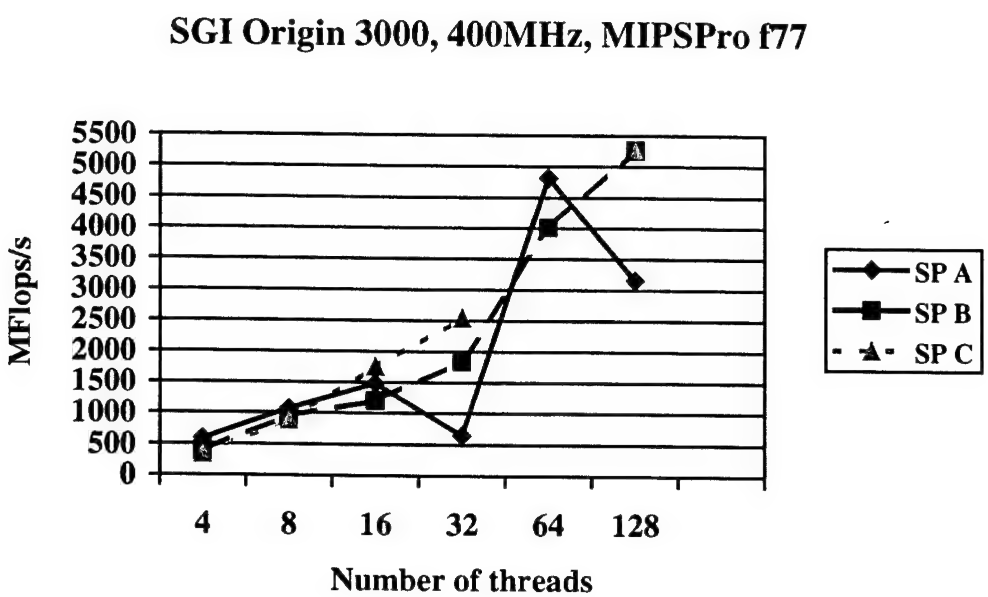

Figure 14. PBN SP benchmark on an SGI Origin 3000 with the vendor compiler.

6. References

1. OpenMP website. <http://www.openmp.org/>.
2. Bull, M. J. "Measuring Synchronization and Scheduling Overheads on OpenMP." Proceedings of the First European Workshop on OpenMP (EWOMP '99), Lund, Sweden, 1999.
3. Browne, S., and J. Cownie. "OpenMP Debugging With TotalView." Proceedings of the Workshop on OpenMP Applications and Tools, July 2000.
4. Browne, S., J. J. Dongarra, N. Garner, G. Ho, and P. Mucci. "A Portable Programming Interface for Performance Evaluation on Modern Processors." *International Journal of High Performance Computing Applications*, vol. 14, no. 3, pp. 189-204, 2000.
5. Browne, S., J. J. Dongarra, N. Garner, K. London, and P. Mucci. "A Scalable Cross-Platform Infrastructure for Application Performance Optimization Using Hardware Counters." Proceedings of SC'2000, Dallas, TX, November 2000.

INTENTIONALLY LEFT BLANK.

NO. OF COPIES	ORGANIZATION
2	DEFENSE TECHNICAL INFORMATION CENTER DTIC OCA 8725 JOHN J KINGMAN RD STE 0944 FT BELVOIR VA 22060-6218
1	HQDA DAMO FDT 400 ARMY PENTAGON WASHINGTON DC 20310-0460
1	OSD OUSD(A&T)/ODDR&E(R) DR R J TREW 3800 DEFENSE PENTAGON WASHINGTON DC 20301-3800
1	COMMANDING GENERAL US ARMY MATERIEL CMD AMCRDA TF 5001 EISENHOWER AVE ALEXANDRIA VA 22333-0001
1	INST FOR ADVNCD TCHNLGY THE UNIV OF TEXAS AT AUSTIN 3925 W BRAKER LN STE 400 AUSTIN TX 78759-5316
1	US MILITARY ACADEMY MATH SCI CTR EXCELLENCE MADN MATH THAYER HALL WEST POINT NY 10996-1786
1	DIRECTOR US ARMY RESEARCH LAB AMSRL D DR D SMITH 2800 POWDER MILL RD ADELPHI MD 20783-1197
1	DIRECTOR US ARMY RESEARCH LAB AMSRL CI AI R 2800 POWDER MILL RD ADELPHI MD 20783-1197

NO. OF COPIES	ORGANIZATION
3	DIRECTOR US ARMY RESEARCH LAB AMSRL CI LL 2800 POWDER MILL RD ADELPHI MD 20783-1197
3	DIRECTOR US ARMY RESEARCH LAB AMSRL CI IS T 2800 POWDER MILL RD ADELPHI MD 20783-1197

ABERDEEN PROVING GROUND

NO. OF COPIES	ORGANIZATION
2	DIR USARL AMSRL CI LP (BLDG 305)

NO. OF COPIES	ORGANIZATION
1	HPCMO C HENRY PRGM DIR 1010 N GLEBE RD STE 510 ARLINGTON VA 22201
1	HPCMO L DAVIS DPTY PRGM DIR 1010 N GLEBE RD STE 510 ARLINGTON VA 22201
1	HPCMO V THOMAS DISTRIB CTRS PRJT OFCR 1010 N GLEBE RD STE 510 ARLINGTON VA 22201
1	HPCMO J BAIRD HPC CTRS PRJT MGR 1010 N GLEBE RD STE 510 ARLINGTON VA 22201
1	HPCMO L PERKINS CHSSI PRJT MGR 1010 N GLEBE RD STE 510 ARLINGTON VA 22201
1	RICE UNIVERSITY M BEHR MECHL ENGNRG MTRLS SCI 6100 MAIN ST MS 321 HOUSTON TX 77005
1	RICE UNIVERSITY T TEZDUYAR MECL ENGRG MTRLS SCI 6100 MAIN ST MS 321 HOUSTON TX 77005
1	J OSBURN CODE 5594 4555 OVERLOOK RD BLDG A49 RM 15 WASHINGTON DC 20375-5340

NO. OF COPIES	ORGANIZATION
1	NAVAL RSRCH LAB J BORIS CODE 6400 4555 OVERLOOK AVE SW WASHINGTON DC 20375-5344
1	NAVAL RSRCH LAB D PAPACONSTANTOPOULOS CODE 6390 WASHINGTON DC 20375-5000
1	NAVAL RSRCH LAB G HEBURN RSRCH OCNGRPHR CNMOC BLDG 1020 RM 178 STENNIS SPACE CTR MS 39529
1	AIR FORCE RSRCH LAB DEHE R PETERKIN 3550 ABERDEEN AVE SE KIRTLAND AFB NM 87117-5776
1	AIR FORCE RSRCH LAB INFO DIRCTRT R W LINDERMAN 26 ELECTRONIC PKWY ROME NY 13441-4514
1	R A WASILAUSKY SPAWARSYSCEN D4402 BLDG 33 RM 0071A 53560 HULL ST SAN DIEGO CA 92152-5001
1	USAE WTRWYS EXPRMNT STA CEWES HV C J P HOLLAND 3909 HALLS FERRY RD VICKSBURG MS 39180-6199
1	USA CECOM RDEC AMSEL RD C2 B S PERLMAN FT MONMOUTH NJ 07703
1	SPACE AND NVL WRFR SYS CTR K BROMLEY CODE D7305 BLDG 606 RM 325 53140 SYSTEMS ST SAN DIEGO CA 92152-5001

NO. OF COPIES	ORGANIZATION
3	USA HPCRC B BRYAN P MUZIO V KUMAR 1200 WASHINGTON AVE S MINNEAPOLIS MN 55415
1	USA HPCRC G V CANDLER 1200 WASHINGTON AVE S MINNEAPOLIS MN 55415
1	NCCOSC L PARNELL NCCOSC RDTE DIV D3603 49490 LASSING RD SAN DIEGO CA 92152-6148
1	UNIVERSITY OF TENNESSEE S MOORE INNOVATIVE COMPUTER LAB 1122 VOLUNTEER BLVD STE 203 KNOXVILLE TN 37996-3450
1	SDSC UNIV OF CA SAN DIEGO A SNAVELY 9500 GILMAN DR LA JOLLA CA 92093-0505
1	NCSA 152 CAB S SAARINEN 605 E SPRINGFIELD AVE CHAMPAIGN IL 61820
1	USA ERDC D DUFFY CMPTTNL MGRTN GRP MAJOR SHARED RESRC CTR VICKSBURG MS 39180
1	USA ERDC J HENSLEY CMPTTNL MGRTN GRP MAJOR SHARED RESRC CTR VICKSBURG MS 39180
1	USA ERDC M FAHEY CMPTTNL MGRTN GRP MAJOR SHARED RESRC CTR VICKSBURG MS 39180

NO. OF COPIES	ORGANIZATION
1	USA ERDC T OPPE CMPTTNL MGRTN GRP MAJOR SHARED RESRC CTR VICKSBURG MS 39180
1	USA ERDC W WARD CMPTTNL MGRTN GRP MAJOR SHARED RESRC CTR VICKSBURG MS 39180
1	USA ERDC R ALTER CMPTTNL MGRTN GRP MAJOR SHARED RESRC CTR VICKSBURG MS 39180

ABERDEEN PROVING GROUND

NO. OF COPIES	ORGANIZATION
20	DIR USARL AMSRL CI N RADHAKRISHNAN AMSRL CI H C NIETUBICZ S THOMPSON AMSRL CI HC P CHUNG J CLARKE D GROVE D HISLEY M HURLEY A MARK D PRESSEL R NAMBURU D SHIRES R VALISETTY C ZOLTANI AMSRL CI HI A PRESSLEY AMSRL CI HS D BROWN T KENDALL P MATTHEWS K SMITH R PRABHAKARAN

INTENTIONALLY LEFT BLANK.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	April 2002	1 Oct 2000–1 Jun 2001

4. TITLE AND SUBTITLE
Reality Check on OpenMP Implementations

5. FUNDING NUMBERS
665803.731

6. AUTHOR(S)
Shirley Moore,* Daniel Pressel, and Juan Carlos Chaves †

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
U.S. Army Research Laboratory
ATTN: AMSRL-CI-HC
Aberdeen Proving Ground, MD 21005-5067

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-TR-2718

9. SPONSORING/MONITORING AGENCY NAMES(S) AND ADDRESS(ES)

10.SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
* University of Tennessee, Knoxville, TN 37996
† HPTi/Major Shared Resource Center, U.S. Army Research Laboratory, Aberdeen Proving Ground, MD 21005

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*
OpenMP is a proposed industry standard Application Programmer Interface (API) that supports shared-memory parallel programming in Fortran and C/C++ on architectures including Unix, Linux, and Windows NT platforms. This report discusses experiences using OpenMP implementations on Shared Resource Center (SRC) platforms. The experiences include running OpenMP benchmarks, as well as using OpenMP with applications. Tools available for debugging and analyzing OpenMP programs are also covered. Most of the results in this report should be considered preliminary and the basis for further investigation.

14. SUBJECT TERMS
benchmarking, OpenMP, supercomputing

15. NUMBER OF PAGES
22

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

NSN 7540-01-280-5500
Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18 298-102

INTENTIONALLY LEFT BLANK.